Задача A. Распределенные вычисления

Ограничения задачи позволяли явно просимулировать процесс распределения наборов данных между серверами.

Запустим цикл и будем поддерживать номер текущего сервера, которому необходимо выдать наборы данных для обучения. Цикл необходимо завершить в тот момент, когда мощность текущего сервера больше, чем количество оставшихся наборов данных.

Асимптотика такого решения — $O(n)$.

Задача B. Сжатый пароль

В задаче требовалось удалить из строки не более k символов так, чтобы в строке, которая получится после удалений, было минимальное количество различных символов. То есть наша задача — выкинуть из пароля все вхождения как можно большего количества различных букв.

Для каждой буквы найдем, сколько раз она встречается в пароле. Заметим, что выгоднее удалять буквы, которые встречаются реже других. Действительно, пусть пусть буква «a» встречается c_a раз, буква «b» — c_b раз и $c_a < c_b$. Чтобы уменьшить ответ на 1 нужно удалить все вхождения «a» или «b», но нам выгоднее удалить меньше символов.

Таким образом, в решении необходимо для каждой буквы посчитать количество ее вхождений в пароль. После этого нужно неявно удалять из пароля самые редко встречаемые буквы до тех пор, пока суммарное количество удаляемых символов не превышает k.

Сложность такого решения — $O(n)$.

Задача C. Детектор ошибок

Перед обработкой периодов, которые интересуют Колю, предподсчитаем массивы `up` и `down`. В массив `up` запишем такие номера моментов i, после которых количество ошибок увеличивается, то есть $a_i < a_{i+1}$. В массив `down` запишем номера моментов i, после которых количество ошибок уменьшается, то есть $a_i > a_{i+1}$.

Заметим, что период с l по r является сбоем, если все моменты i ($l \leqslant i < r$), для которых $a_i < a_{i+1}$, происходят до моментов j ($l \leqslant j < r$), для которых $a_j > a_{j+1}$. Иными словами, все элементы массива `up`, которые лежат в отрезке $[l, r-1]$, меньше любого элемента массива `down` из того же отрезка.

Это условие можно проверить при помощи бинарного поиска: найдем наибольший элемент `up`, меньший r, и найдем наименьший элемент `down`, больший либо равный l, и сравним найденные элементы. Если элемент массива `up` меньше элемента массива `down`, то данный период — сбой.

Так же стоит отдельно рассмотреть случай, когда количество ошибок в течение всех изменений неубывает или невозрастает. В таком случае массивы `up` или `down` будут пустыми.

Временная сложность такого решения — $O(n + m \log n)$.

Задача D. Безопасный код

Заметим, что произведение элементов массива будет меньше, если оно будет отрицательным. А чтобы произведение метрик было отрицательным, среди значений не должно быть 0 и количество отрицательных метрик должно быть нечетным.

Будем действовать следующим образом, жадно улучшая результат на каждом шаге.

Если среди значений метрик есть нули, будем прибавлять или вычитать d из метрики, равной нулю. При этом необходимо поддерживать количество отрицательных значений нечетным. Если количество нулей больше k, то мы никак не можем получить результат, отличный от нуля, и любая последовательность действий будет правильным ответом. Иначе, чтобы получить отрицательное произведение, надо изменить каждый ноль как минимум один раз (и при этом мы всегда можем получить отрицательное произведение).

Если текущее произведение метрик положительно, то мы хотим изменить знак значения одной метрики. В качестве такого числа надо брать наименьшее по модулю. Предположим, что у нас есть две метрики x и y, $|x| < |y|$. Докажем, что менять знак y не выгодно. Пусть m — минимальное число операций, необходимое для изменения знака y. Если мы применим m операций к y, то модуль x не изменится, а модуль y станет равен $d \cdot m - |y|$. Если же мы применим m операций к x, то

модуль числа x станет равен $d \cdot m - |x|$, модуль y не изменится. После проделанных операций произведение будет отрицательным, поэтому необходимо максимизировать произведение модулей. $|y| \cdot (d \cdot m - |x|) > |x| \cdot (d \cdot m - |y|)$, так как $|y| > |x|$ и $d \cdot m - |x| > d \cdot m - |y|$.

После того, как произведение метрик стало отрицательным, следует выбирать минимальное по модулю число и увеличивать его модуль. Пусть есть две метрики со значениями x, y и еще сколько-то метрик с отрицательным произведением p. Если $|x| < |y|$, то $(|x|+d) \cdot |y| \cdot p < |x| \cdot (|y|+d) \cdot p$. Таким образом выгодно увеличивать (или уменьшать, когда она отрицательная) метрику с минимальным значением по модулю.

Будем повторять описанную выше операцию до тех пор, пока мы не исчерпаем лимит в k изменений.

Для быстрого поиска элемента с минимальным модулем следует использовать приоритетную очередь (`priority_queue` в C++) или множество (`set`) на основе дерева поиска. Тогда время работы решения будет $O(n + m \log n)$.

Задача E. Локальная сеть

Описанную в задаче схему сети можно представить в виде ориентированного графа. В этом графе сервера и компьютеры сотрудников буду вершинами, а подключения — ориентированными ребрами. Задача заключалась в построении графа, в котором существует путь от любого сервера до любого компьютера, не проходящий через другие компьютеры (так как по условию соединений компьютера с компьютером напрямую быть не может). Среди всех таких графов требуется найти граф с минимальным количеством ребер, а среди всех таких — с минимальной суммарной квадратов декартовых расстояний между вершинами.

Заметим, что в таком графе должно быть хотя бы $n + m - 1$ ребро. Действительно, неориентированная версия такого графа должна быть связной, а в связном графе из $n + m$ вершин не может быть меньше ребер. Заметим, что тогда в этом графе нет циклов, а значит если из a достижима b, то из b не достижима a. В таком случае все компьютеры должны быть напрямую подключены к одному из серверов, с которым связаны все остальные сервера.

Таким образом, минимальное количество ребер достигается, когда все сервера пересылают сообщения какому-то одному серверу (пусть это будет сервер x), а он отправляет их всем компьютерам. Схему передачи сообщений к серверу x можно представить в виде подвешенного дерева. Суммарная длина подключений в таком случае равна сумме расстояний от сервера x до всех компьютеров и сумме длин ребер в дереве подключений между серверами. Эти два слагаемых можно оптимизировать независимо.

Давайте минимизируем суммарную длину связей в дереве подключений между серверами. Эту часть задачи можно решить алгоритмом поиска минимального остовного дерева в полном графе подключений между m серверами, для чего удобно воспользоваться алгоритмом Прима с асимптотикой $O(m^2)$. Найти минимальную сумму расстояний от сервера x до всех компьютеров сотрудников можно, перебрав все сервера и для каждого посчитав расстояние до всех компьютеров за $\mathcal{O}(nm)$. Так можно сделать, поскольку выбор сервера x влияет только на направление подключений в минимальном остовном дереве серверов.

Таким образом, задача сводится к построению минимального остовного дерева за $O(m^2)$, перебору всех серверов и выбору среди них сервера, с минимальным суммарным расстоянием до компьютеров за $O(nm)$. Общее время работы такого решения — $O(m \cdot (n + m))$.